Massachusetts Institute of Technology
Artificial Intelligence Laboratory

A.I. Memo 523

Logo Memo 52
May 1979

Logo Music Projects:

Experiments in Musical Perception and Design

by

Jeanne Bamberger

ABSTRACT

This memo gives a series of experiments which one can use to get a better understanding of how music works and how music is apprehended by an active and knowing listener. It does so by using the children's computer language, LOGO, and capitalizes on the use of procedural thinking and other programming concepts (for example, the use of variables) in the designing and analysis of melody and rhythm.

The work reported in this paper was supported in part by the National Science Foundation (under grant number 77-19083SEI) and conducted at the Artificial Intelligence Laboratory and the Division for Study and Research in Education, Massachusetts Institute of Technology, Cambridge, Massachusetts. The views and conclusions contained in this paper are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the National Science Foundation or the United States Government.

TABLE OF CONTENTS

LOGO MUSIC--SOME HOW TO'S

This memo is intended for people in the Logo Lab who would like to use the system to learn more about how music works. What follows assumes a working knowledge of LOGO, no knowledge of music and an interst in thinking about music procedurally.

I

The first (and by now classical) game is called TUNEBLOCKS. There are two versions of TUNEBLOCKS, both of them designed to involve their players in such issues as:

A. Thinking about a melody as a procedure.
The hierarchic structure of simple melodies.
A melody as a model of modular structure.

B. How you represent to yourself a particular aggregate of pitch-time events, i.e. a tuneblock.
What are salient features (for you) and can these change?
How can you discriminate one tuneblock from another?

C. What gives a tune its particular coherence?
What generates a particular melodic function (begin, end)?
The influence of context (and contextual embedding) on the "same" set of pitches and durations.
Melodic transformation as a compositional tool.

TUNEBLOCKS I

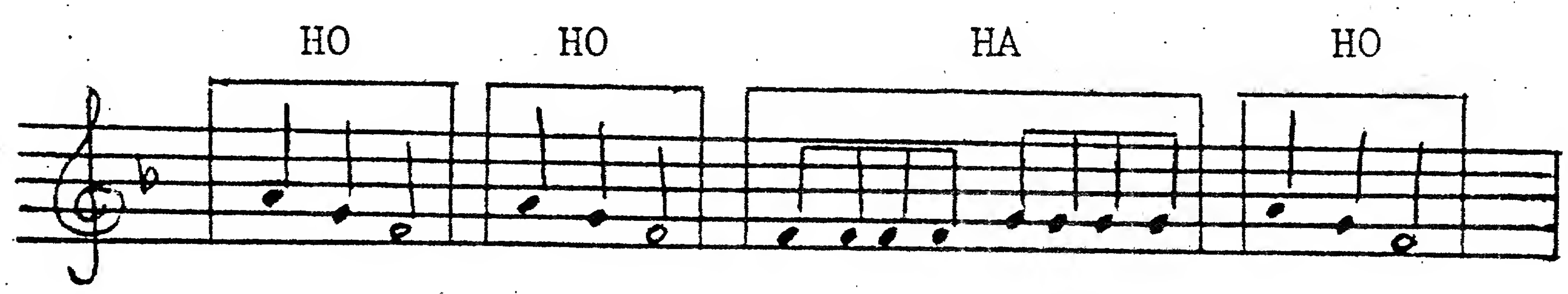
In this version of the game, you begin by listening to a whole tune. Going down one level in the structural hierarchy, you then listen to the blocks from which this whole tune is made. A block can be thought of as a relatively contained entity--a module of the larger structure, or as the smallest "meaningful" chunk of the whole. More specifically, when an aggregate of pitch-time events is used more than once in a tune, it is always considered a block. While all blocks are not used more than once, those that are, help to define the boundaries of those that are not. A block, then, is a meaningful subprocedure of the whole.

The procedures you use for playing TUNEBLOCKS are as important to think about as the tune itself, as a procedure. For example, the name of a block (S1, HO or RED) has no descriptive meaning in itself--i.e., it doesn't tell you anything about the "contents" of the block, its characteristic features--it is merely a label that will come to stand for whatever you make it stand for. The meaning you assign to that label (or that you associate with that label) you may express in a picture, some words or a gesture. These discriptions will tell you something about what you have captured as the significant features of the block. And as you continue working with the blocks, new or even different features may emerge.

Strategies for reconstructing the tune are also significant to your way of thinking-a-tune. For example, you may find yourself simply looking for the block with which the tune begins, and then next-next-next. However, this isn't always so obvious a ploy--context may fool you. When you discover that the "same" block is surprisingly no longer the "same," you need to wonder why. Or you may find it more natural to think of the tune in bigger chunks, e.g. you may want to build larger modules made up of the smaller tuneblocks, for example, a beginning, a middle, and an end, rather than simply next-next-next. The strategy you choose can tell you soemthing about how you listen.

TUNEBLOCKS I, then, provides you with a mini-world for actively taking apart and putting together a coherent musical structure. Perhaps this version of the game is most interesting in its focus on what is an element, where are significant boundaries, how do bounded entities aggregate, etc. With young children, this often becomes a central issue along with that of "functional naming." This latter takes several forms:

1. A block which functions to begin a tune often seems to demand a different name when it functions to end a tune. Thus, in "HOT CROSS BUNS," children find it very uncomfortable, even resist (or conversely submit with a sense of magic) using the same name for the first and last blocks:



Indeed, the HO at the end has a different function from the HO's at the beginning and, indeed, it sounds different when it follows HA. How, then, can it have the same name? So, if you understand the "meaning" of HO to be a certain set of pitch-time events, it is the "same" at beginning and end. But if you understand the "meaning" of HO to be "the first part" or "the beginning," you need a different word when it's "the last one" or "the end."

2. The boundary question comes in several guises, too. For example, is the "first part" of "HOT CROSS BUNS" HO or is it HO HO? Children will frequently think that one HO is sufficient. They seem to be saying, "The tune begins with that thing;" the name seems to stand for a "kind of stuff." That the procedure, HO, is only one instance of this kind of stuff, that doing it again (i.e., two instances of the thing) makes up "the first part," has to be discovered. In the process, the meaning of HO again takes on new significance--we might say it becomes more like a set of properties rather than a place marker. So just what is "an element" is a continuing issue in this mini-world.

Most important, all of these ways of thinking-a-tune and its part are valid, significant, right! The point is to really take all of them quite seriously because each one reveals a significant aspect of the tune, how we come to understand it, to describe it and to remember it. In turn, each way of representing the tune provides a potential tool for developing sophisticated procedural means of analysis and composition as well as notions that may bear on the performance of music and musical taste. Finally, what appear here to be pretty much domain-specific issues can stretch out to include issues that cut across domains--like naming, modes of description, modular programming, context, function, etc.

TUNEBLOCKS II

In this version of the game you start with just a set of tuneblocks. The goal is to use them so as to create a whole tune that you like. Once again your procedure is at least as important as your finished product. In fact, your modes of description, the features you capture in your descriptions of the blocks and the relations you find among them will strongly influence your finished product. In playing this version of the game, both children and adults often find it useful to make a "Block Dictionary." The dictionary consists of an "entry" for each of the blocks with which you are working, along with a list of all the features you have noticed about each one. The list of features for each entry grow and may also change as you go along. This is a particularly interesting process when you are working with a set of blocks which may initially be quite strange to you--i.e., you can't, at first, conceive of a "sensible tune" which they might make. Or putting it another way, your model of a sensible tune does not include this kind of musical material. In this situation, you may discover a new coherence in music composed in a style that is unfamiliar to you.

When you have completed your own tune, it is always interesting to listen to the original tune from which the blocks were taken. But remember, your purpose is not to arrive at the original tune--indeed, many players insist that they like their own tune better than the original. It's useful to think about the differences between possible tunes made up of the same blocks: Why do you like one "solution" better than another; what are the differences in the structural relations within each tune; how has the playing procedure adopted by various individuals influenced their final product; how have the modes of representing blocks influenced both the procedure and the final product? (For a discussion of this whole process see: Bamberger, "In Search of A Tune" in The Arts and Cognition, ed., Perkins and Leondar.)

TUNEBLOCKS WITH AMB

Two Students Talk About Their Work

I

3 3 3 3 1 3 3 5 3 3 5 1 1 5 3 3 4 3 3 2 4 4 3 3 3 3

Immediately, this set presented itself as a challenge, both for rhythmic grouping and variation, and tonal interest (so it doesn't just go wandering all over), if not for a definite tonal center. I wanted the last note of each block except 5 to be longer. No block was seen to be a potential ending; in fact, no combination seemed to make any more sense than any other.

Initial strategy was to try to find <u>some</u> rules, at least, to develop some sensible sequence. Finding features was tough--there was no rhythmic pattern, no pitch shape to speak of, no harmonic structure. I felt I was using only a sketchy tonal representation, each block seen as perhaps one note. Attempts to latch onto features other than this proved fruitless.

So I made my own. Repeating a good "moving" block, 3, four times, introduced a regularity and served to motivate the piece by virtue of <u>skipping a beat each time</u>. The result was an African drum motif which continues as an undercurrent to the whole piece. Any change from this "beat" is very noticeable, so it must be delicately handled at first, one block at a time, always restating the beat on both sides. I tried to force on these blocks the feeling that more than one sequential line of music was happening at once.

Block 3 sets the stage. 1 is an attempt to break out of the "beat." It is frustrated by the return of 3-3. <u>Then 5, a longer version of 1</u>, with the same result, setting 3 up as a very strong force. Finally, a break out of the mold, 5-1-1-5, very assertive because of the adjacent 1's and the break of the beat twice by 5. But then the beat (3) reutrns, played off

against the similar block 4. Somehow the tonal section this time, 2-4-4, does not require a concluding 2, perhaps because 3 has lost some of its strength through repeated intervening of other blocks. These two groups are seen as definite tonal assertions within a sea of droning rhythm. The concluding 5 is needed to make sure the group is heard in its full, fist-shaking strength before reaffirmation of the beat motif. However, 2-4-4 is strong enough (perhaps it is somehow more lyrical) in relation to the beat. Interestingly enough, by the time we hear this group it sounds like its "attempt" (4-3-3) was actually 2-3-3, paralleling 5-3-3-5-1-1-5.

The ending, simply a further reaffirmation of 3, is sort of a fade-out which can be heard to continue on, perhaps repeating the whole as it started.

II

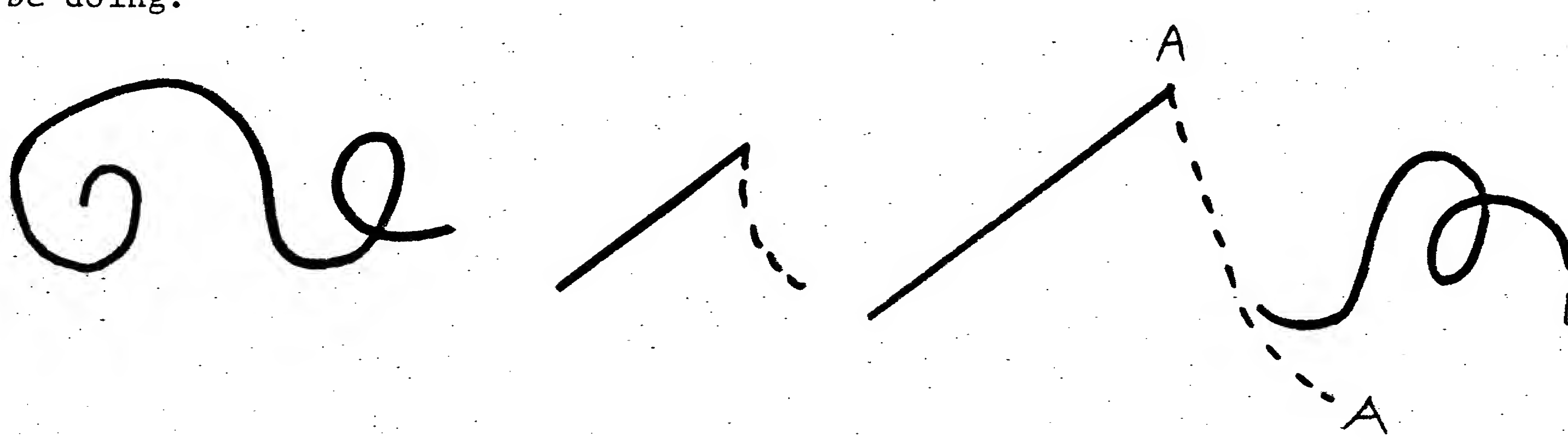
What I did.

I listened to each of the blocks, notating the pitches, then to the string 1-2-3-4-5. The observation that all of the notes were of equal duration, that all of the pitches fit within one octave of a white note scale, the step-wise notion of the lines all suggested that these might be pieces of Gregorian chant. If this is so, certain assumptions may be made. The range (A-A) and the distribution of pitches (there are many more D's than anything else, and all but one of the blocks end on D) allow one to classify the mode (mode II, Hypodorian), which makes it reasonably safe to assume that the piece will end on D. The important things which I know about (which I have seen in) Gregorian melodies, though,

is their almost universal arched shape--that is, from a lower pitch level the melody moves to a peak somewhat after the mid-point of the melody, and returns--and the extreme care taken to control the use of leaps and the extremes of the range. Working entirely from my notation, I made some decisions. It seemed clear that 4, with its circling and then repetition upon settling on the final of the mode, would be the most satisfactory ending. I was concerned about the 2 blocks beginning on G, since at least one of them could not be at the beginning and would result in the leap of a fourth. I then did all the necessary juggling in my head to make a tune that made sense to me.

My tune: 2-4-1-5-3-4

This tune is in three sections: 2-4, 1-5, 3-4. I find that the first section is very smooth, and outlines its space of a 4th in a very nice way. With D as its starting point and center, it moves gently up to F, turns around D, and picks up the C. The next section begins boldly, leaping immediately to G, a new note, but returns quickly and gracefully to D, the resting point. This TIME, with the G already prepared, it pushes to the high A which is obviously the climax of the piece. Then "musical gravity" seems to operate, and balance is restored to me in the third section by the low A, winding down with a little circle around D. This drawing is an attempt to represent what I hear the piece to be doing.



In fact, this could probably represent the activity--the "purpose" of a great many gregorian melodies.

***

Both students are MIT undergraduates. Student I, a math major, had no previous musical training before this experience with tuneblocks. Student II, a music major, is a pianist and has taken lots of music theory courses. The two strategies and the resulting tunes are remarkable in their differences especially as they illustrate the influence of beginning assumptions.

BASIC MUSIC PLAYING PROCEDURES

The file, INIT, contains the basic procedures you will need for all the music projects. For convenience, when you LOGIN: KIDS, the file INIT will be automatically put into your workspace.

1. <u>NOTE :P :D</u>

NOTE is the only music "primitive"--i.e., it is the single procedure from which all the others are built. With this procedure you can describe just one note. A note has two properties, pitch and duration. Thus, in order to create a note you must describe both its pitch and the duration of that pitch. Thus, the command, NOTE, requires two <u>inputs</u>: :P indicates the pitch; :D indicates the duration. To use NOTE you must give :P a <u>value</u> which indicates exactly which pitch you want and also give :D a <u>value</u> which indicates exactly which duration you want that pitch to have. The following table shows you how numbers have been assigned to pitch:

-24	-23	-22	....	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	....	35
C_2	$C\#_2$	D_2		A_1	$B^{\flat}_1$	B_1	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C^1		B^2

Thus, the lowest pitch is -24 (C_2) and the highest is 35 (B^2), a range covering 5 octaves. The number 0 has been assigned to middle C. The positive integers name the chromatic pitches going up from middle C, the negative integers name the chromatic pitches going down from middle C.

Duration is also indicated by numbers. Larger numbers indicate longer durations; smaller numbers indicate shorter durations. The values of durations are proportional to one another; thus a duration of 4 is twice as long as a duration of 2 and half as long as a duration of 8. You can use durations from 1 to 250; usually durations range from 1 to 24. For example:

NOTE 0 2	Middle C for a duration of 2
NOTE 1 2	C# for a duration of 2
NOTE 2 5	D for a duration of 5
NOTE 3 3	D# for a duration of 3

2. PLAY [:PITCH] [:DURS]

It is easier and more natural to think about and describe a melody as a whole set of pitches (a string of pitches) and its associated set of durations (a string of durations). Thus, when describing a melody, it is more convenient to use the command, PLAY. PLAY also needs two inputs-- PLAY [:PITCH] [:DURS], but the value you give to :PITCH will be a string of numbers indicating a series of pitches, the value you give to :DURS will be a string of numbers indicating the duration of each of the pitches. The computer will assign the first pitch to the first duration, the second pitch to the second duration, etc. Thus, if you want to try a melody, you can first think about only the pitches and then (while, perhaps clapping the "rhythm" of your tune), describe the durations. For example:

(1) PLAY [0 1 2 3 4 5] [4 4 2 2 4 8] — This will play the chromatic scale from middle C to F with the rhythm, 4 4 2 2 4 8.

(1b) PLAY [0 1 2 3 4 5] [6 6 3 3 6 12] — Notice that the time relations are the same but the whole thing is proportionately slower.

(2) PLAY [0 2 4 5 7 9 11 12] [4 4 2 2 4 8 8 8] — This will play C D E F G A B C with the same rhythm as (1) plus two more 8's.

(3) PLAY [0 -5 -8 -12 12] [2 3 4 5 6] — This will play C G_1 E_1 C_1 C^1 with the rhythm, 2 3 4 5 6.

(4) PLAY [2 2 9 9 11 11 9] [2 2 2 2 2 2 4] — This will play the first phrase of Twinkle Twinkle Little Star.

(5) PLAY [0 0 7 7 9 9 7] [2 2 2 2 2 2 4] So will this.

Notice that in the last two examples the rhythm is the same but also the relations between pitches are the same.

3. PM

So far you have learned only about how pitch and duration (notes) are described in LOGO. Sending such instructions to the computer will cause the computer to compute the results of your instructions but you will not hear those results until you add the command, PM. The command, PM (for Play Music) tells the computer to send the computed results to the music box where they are translated into "hearable" results. Thus, at the end of any music playing command you must add PM (either after a space on the same line or on the next line) in order to hear the results of your instructions.

4. REST

It is useful (and often important) to describe silence. A REST, or silence, is best thought of as if it, too, were a note. Thus, the word REST (or, if you wish, the number -28) can be included in your pitchstring and its duration can be included in your durationstring. For example:

```
PLAY [1 2 3 REST 2 3 4 REST 3 4 5] [2 2 2 2 2 2 2 2 2 2 2]
```

This will play:

PITCH	1	2	3	REST	2	3	4	REST	3	4	5
DURATION	2	2	2	2	2	2	2	2	2	2	2

Notice that the rest (silence, a "gap") causes pitches to group together; that is, the rest functions as a boundary maker, containing or delimiting the events on each side of it. In this example the rests help to reinforce the pitch sequence. A sequence is the repetition of a small figure on

successively higher or lower pitches. Thus we have:

```
1 2 3    Initial figure
2 3 4  }
         } Sequential repetition
3 4 5  }
```

Can you imagine how you might write a procedure to generate sequences for you?

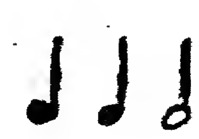
5. <u>BOOM [:DUR] and SSH [:DUR]</u>

BOOM and SSH are percussion sounds which can be used to experiment with rhythms and later to make accompaniments for tunes (see DRUMMER and DRUM).

BOOM and SSH take just one input, a duration or a string of durations. For example:

BOOM [4 4 4 6 2 4 4 4 4 6 2 4]	Plays just the rhythm of the first two phrases of "America" on the drum.
BOOM [2 2 2 3 1 2 2 2 2 3 1 2]	So does this, but the <u>tempo</u> is faster--i.e., the basic beat is 2 instead of 4, but the time <u>relations</u> are the same.
BOOM [6 2 4 4 4 4 8 4 4 8 4 4 8]	This plays just the rhythm of the first part of "Mary Had a Little Lamb."
SSH [2 2 2 2] BOOM [4 4]	This plays a "cha cha" rhythm with a brushed cymbal sound followed by the drum sound.

You can also use BOOM to play just the rhythm of a TUNEBLOCK on the drum.

BOOM "HO	Plays  on the drum.
SSH "HA	Plays on the brushed cymbal.

6. REPEAT [:THING] :TIMES

REPEAT allows you to REPEAT a procedure [:THING] a given number of times [:TIMES].

REPEAT [HO] 3	Plays the block, HO, three times.
REPEAT [BOOM [2 2 4] 3]	Plays BOOM [2 2 4, 2 2 4, 2 2 4]

7. TOG :TUNE1 :TUNE2

This procedure makes it easy to play two tunes or any two procedures simultaneously.

TOG "HOT "STAR	The tunes, HOT and STAR will be played togehter.

You can use TOG to try drum accompaniments for tunes (see also DRUM). For example:

```
TO ACCOMP :TIMES
10 REPEAT [[BOOM 2 2 4] :TIMES]
END
```

```
TOG [ACCOMP 8] "HOT
```

This will play BOOM [2 2 4] eight times together with the tune HOT.

```
TOG [ACCOMP 24] "STAR
```

This will play the same rhythm repeated 24 times together with the tune, STAR.

DRUMMER

The computer music box includes two percussion sounds--one sounds like a tom tom drum, the other like a brushed cymbal. You can begin to experiment with simple rhythms by working first with procedures that do not require you to measure the time of each drum or brush hit. The following is a list of possibilities. Each letter or sign plays one BOOM on the drum or one SSH on the cymbal. The time value of each is given in the column on the right:

1.	S	-	SMALL BOOM	(BOOM 2)
2.	B	-	BIGGER BOOM	(BOOM 4)
3.	@	-	BIGGEST BOOM	(BOOM 8)
4.	T	-	TINY SSH	(SSH 2)
5.	Z	-	BIGGER SSH	(SSH 4)
6.	&	-	BIGGEST SSH	(SSH 8)
7.	%	-	REST	(REST 4)

With these procedures you can simply type the letter names all in a row (with a space between each) to make up rhythms and explore their possibilities. At the same time, you can begin to get a sense of how the time values will work so that later you can make up more interesting combinations of BOOM's and SSH's. To begin, try:

S S S S S   IIIII

B B B B B   I I I I I

Now, how would you make the "Lone Ranger" rhythm (otherwise known as the Overture to William Tell)?

S S B S S B S S B B B

How about the rhythm for "Shave and a haircut--six bits."

B T T B B Z B B

To play this same rhythm using BOOM and SSH, you would have to do the following:

```
BOOM 4
SSH[2 2]
BOOM[4 4]
SSH 4
BOOM[4 4]
```

Or you could play it with a different "instrumentation:"

```
B S S B B % B B
```

That would be easier to do with BOOM:

```
BOOM [4 2 2 4 8 4 4]
```

Notice that BOOM 8, in the last example, sounds just the same as B% or BOOM 4, REST 4. Why?

DRUM

Almost everyone is able to "keep time" to a tune. Even very young children seem to be able to sense an underlying pulse and generate it bodily. While the people process may seem mysterious, it's not hard to make the computer drum act like a person who is keeping a steady beat that "fits" with a tune. Two dimensions of the accompaniment must be considered: 1) The rate of the underlying pulse or beat (the "unit time") and 2) the total number of beats necessary to "fill up" the whole tune ("total time").

The game begins with the people process:

a) Ask the computer music box to play a tune--perhaps the one you have just finished building with tuneblocks.

b) Keep time while you listen--i.e., play a constant beat--hits on a drum or claps which mark off time into invarient units.

c) Listen again and this time count up the number of hits you make from the beginning to the end of the tune.

The goal of the project is to get the computer drum to play what you just played.

Begin by typing DRUM. This allows you and the computer to carry on a conversation. The computer will first ask you WHAT TUNE? You respond with the name of the tune you are working with, let's say, STAR. The computer asks you HOW FAST? On this first try it is good to just guess a number between 2 and 8. Later on, when you want to experiment with the time of the beat to make it fit with the tune, you will need to know that smaller numbers generate a "smaller" beat-time (a faster beat) and larger numbers gen-

erate a "larger" beat-time (a slower beat). Now the computer asks you HOW MANY? You can guess again or you can try the count-up you found in your own performance. The computer will remember all your requests, put the beat together with the tune and play it back for you. Here is a sample conversation:

You:	DRUM	This initializes the conversation.
Computer:	WHAT TUNE?	What tune would you like to work with?
You:	STAR	You want to try STAR.
Computer:	HOW FAST?	How fast would you like the unit time to be? You can guess a number between 2 and 8.
You:	5	Try a beat time of 5.
Computer:	HOW MANY?	How many beats would you like in all?
You:	25	Try 25 beats in all.

Almost certainly the result will be different from your performance; the beat will probably not "fit" in either dimension--i.e., HOW FAST or HOW MANY. But the computer's performance may be more interesting than yours. In fact, the computer accompaniment may be one that you would find hard to play or even to imagine. For some players, this result is more exciting than simply fitting the beat to the piece. But how can you get the computer to make such interesting accompaniments on purpose, not just by chance? And how are such accompaniments different from ones that "fit?" Let's consider the possibilities.

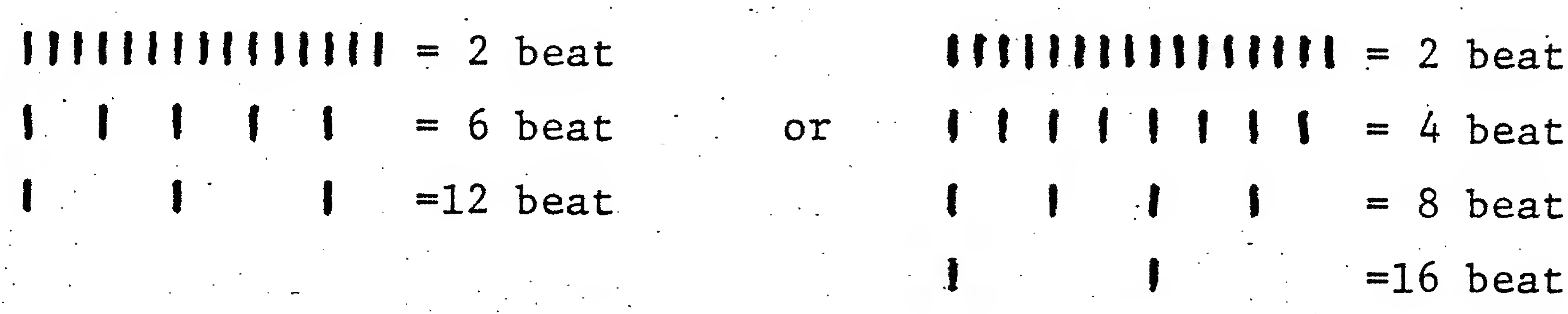
The beat you generated may sound like it's "out of phase" with the tune. That is, if you listen carefully, you will hear that drum hits occur <u>between</u> events of the tune or perhaps coincide at odd places. Or, on a more general

level, you may simply notice that the beat you have generated is too fast or too slow. Actually, it's not always easy to tell which way to go--i.e., whether you need to make the beat slower (use a bigger number for HOW FAST) or faster (use a smaller number for HOW FAST). Try experimenting with the value for HOW FAST and listen to what happens. Just as you can tell when you have a beat that doesn't fit, you will almost certainly be able to tell if you have a beat that <u>does</u> fit. Your experiment should include your own performance of keeping time to the tune--i.e., go back and listen to the tune while you clap the beat, then compare the computer generated beat with your claps. Is the beat you have asked the computer to generate the same as the one you clapped? In fact, there is more than one value for HOW FAST that will fit. We will return to that in a moment.

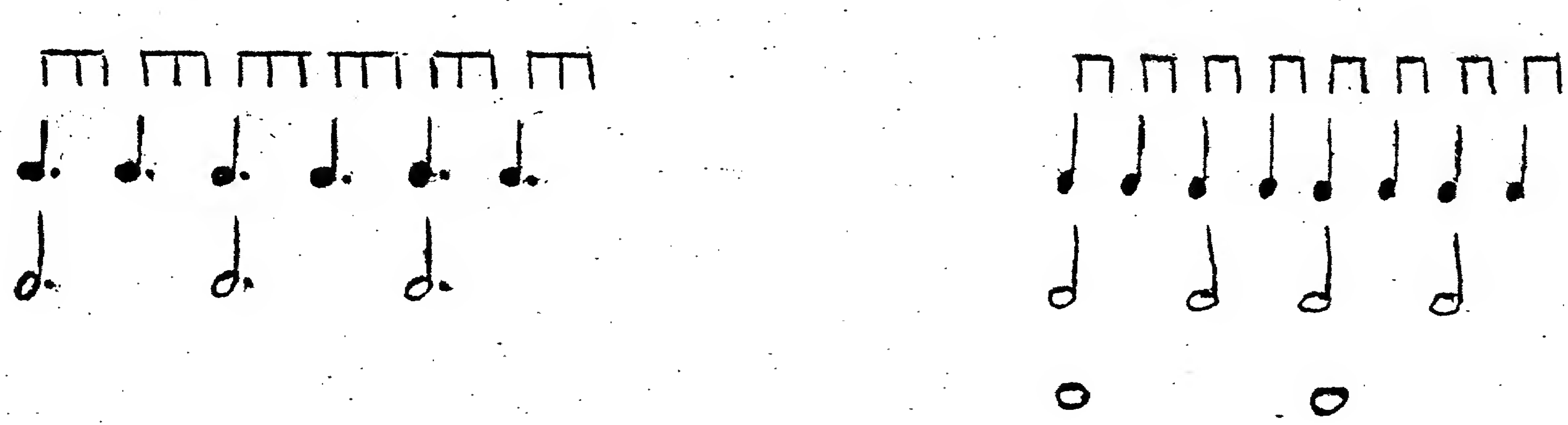
Once you have a beat that fits, you will also notice that the beat doesn't coincide with <u>every</u> event in the tune. This tells you something important about the difference between the "rhythm of the tune" and the underlying beat which it generates: a beat is always a set of hits <u>all of the SAME duration</u>; or, putting it another way, a beat marks off time into <u>equal</u> units. The tune, on the other hand, usually includes a variety of durations. Its events mark off time into <u>varied</u> units. But these varied durations still generate an underlying pulse which you find in tapping your foot and which you recognize when you find the "right" value for HOW FAST. Later on you can use this underlying beat to measure the varied durations of the tune. The beat forms a kind of grid against which you hear and organize the rhythm of the tune. (See TRYS for more on measuring varied durations.)

You might wonder, now, if a beat of 4 (or whatever you found) is the only unit time that will work. Can you clap a slower beat that also fits? Probably you can. Can you hear the relation between the faster and slower beats? Test your hunch. Can you clap a faster beat that fits and test that guess with another value for HOW FAST? Notice that the faster beat divides the 4 (or whatever you have found) while the slower beat <u>groups</u> the 4-beat to form a larger (or slower) beat. The relations among these various beats describe the hierarchy of temporal relations generated by your tune.

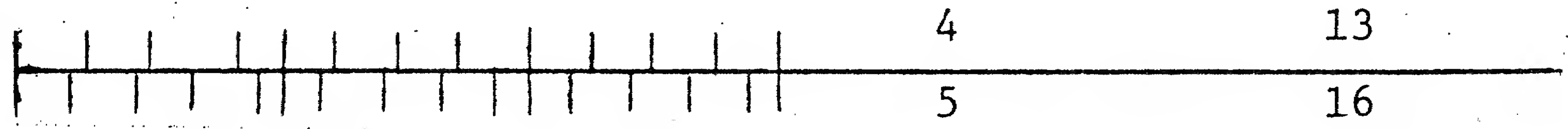
It's often useful in thinking about temporal organization to make a spatial analogue. For example, if you play a fast beat, it might look like this: //////////, while a slower beat might look like this: / / / / /. A picture of the hierarchical relations of the temporal structure might look like this:



In standard music notation this would like like the following:



What about the number of beats? Let's say that 25 came out just about even when you used a unit time of 5. When you tried a beat-time of 4, the beat fit, but did it come out even with 25? Too many or too few? Of course the beat-time and the number are interrelated. A unit time of 4 will require a bigger number for HOW MANY. Think of it like this:



HOW FAST?	HOW MANY?
4	13
5	16

Thus, a smaller unit time takes up less space so you need more of them to make up the same total time. It's like a big kid and a little kid running the same distance in the same time--the little kid has to take more smaller and faster steps, while the big kid takes fewer big and slower steps. It's interesting that young children suggest that the littler kid went farther. It's also interesting that young children will often vary both unit time and number according to the results of their first test. For example, let's say that 25 was too many and 5 was too slow. Well, we can fix both at once: make the number of times 20 and the unit time 4. The result? The unit time fits but there are not enough. Maybe 25 works now for the number of times; why didn't it work before?

Still using DRUM, invent a pattern of durations (a short figure of varied durations) that, when repeated, works as an accompaniment for STAR. Make up several such figures considering the difference in effect of each. Do some make you actually hear STAR differently?

Write up a log of your experiments. You can take several views:

1. What problems did you encounter (not those of the system) which gave you some new <u>musical</u> insight?

2. How would such games work with children? Can you think of other subject matter domains that could be informed by playing with DRUM? Can you think of other related activities for children to do, with or without the computer?

3. What did you learn that you would like to explore further? What did you learn that you could apply to listening, playing or teaching music?

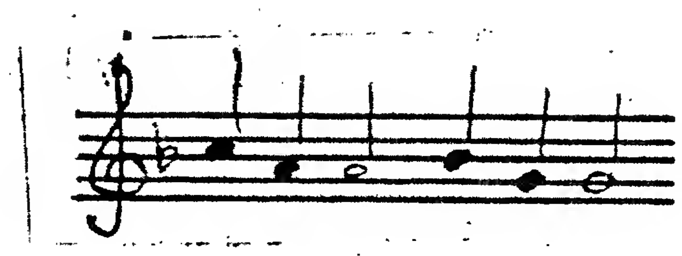
TRYDUR and TRYPITCH are procedures that make it easy to experiment with the durations of a tune (TRYDUR) or the pitches of a tune (TRYPITCH) while keeping one or the other the same. By changing one of the properties of a tune while keeping the other constant, you can study how they influence one another to give a tune its particular effect.

I. TRYDUR :TUNE

TRYDUR (for TRY DURations) makes it possible for you to experiment with the effect of changing the durations of a tune (:TUNE) while the computer keeps its pitches the same. For example, let's say that you want to experiment with the first part of the tune, Gently Row. The first part is called GENTS, so you type:

TRYDUR "GENTS

(1)> ORIG

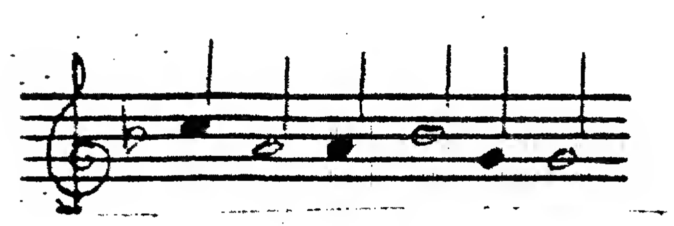


This will play the ORIGinal tune.

(2)> 3 3 3 3 3 3

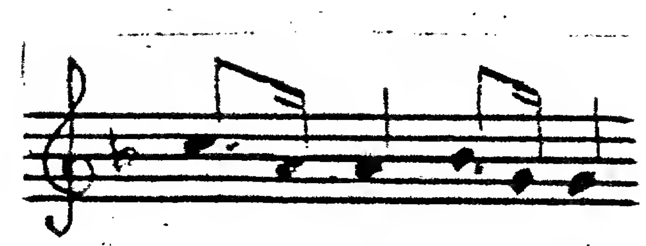
This will play the same string of pitches, giving each a duration of 3.

(3)> 3 6 3 6 3 6



This will play the pitches of GENTS, with the durations 3 6 3 6 3 6.

(4)> 6 2 8 6 2 8



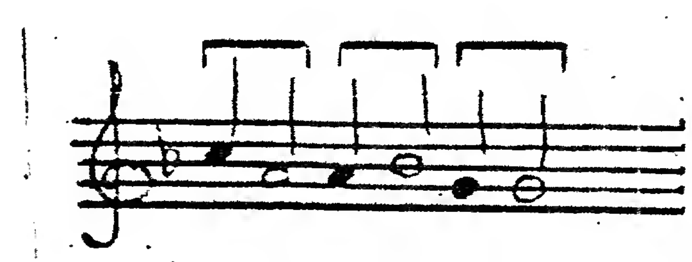
Now you will hear the same pitches again with your requested durations, 6 2 8 6 2 8.

END

This stops TRYDUR and puts you back into LOGO.

Notice that varying the durations while keeping the pitches the same will often make the pitches themselves sound quite different. This is because particular durations will cause the tune events to group or cluster together in new ways. For example, in the ORIGinal, TRY 1, above, you probably hear two groups of three events, each, but in TRY 3, above, you hear three groups of two events, each. This is because a longer duration (6) following a shorter duration (3) tends to create a boundary, a momentary sense of arrival. Thus, in the ORIGinal, you hear ♩ ♩ 𝅗𝅥 ♩ ♩ 𝅗𝅥 , while in TRY 3, you hear ♩ 𝅗𝅥 ♩ 𝅗𝅥 ♩ 𝅗𝅥 . But the difference between these two examples is made even more striking because of the particular pitch relations that result from this change in durations. Notice that in the ORIG we have a sequence--i.e., a figure is repeated twice, the second time a step lower than the first:

This gives the original tune its particular coherence. In TRY 3, that coherence is almost entirely destroyed. Not only is the sequential relationship broken up, but the pitch relations <u>within</u> the new groups are different for each group:



TRY 4, on the other hand, looks like it should be quite different from the original, but, in fact, it sounds quite similar. This is because the changes in duration maintain the original groupings--i.e., you still hear two groups of three notes each, and the sequential relation is also maintained. By making experiments like this, you can begin to discover just what factors tend to create groupings as well as what factors tend to create articulation

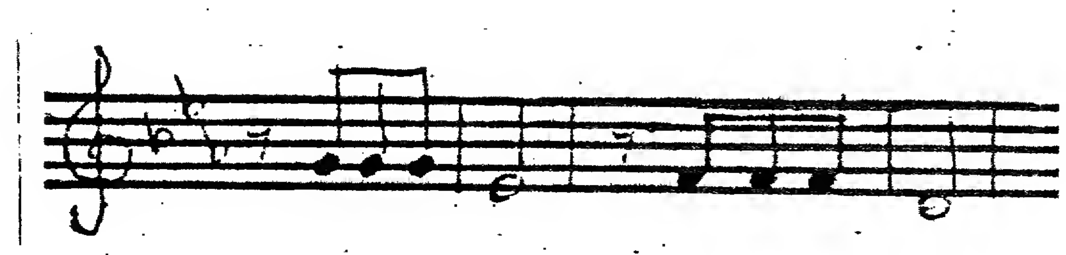
points. In turn, you can learn how to transform a motive so that it maintains its original coherence or how to manipulate durations so that the pitches gain a new coherence. In this way, too, you can gain some insight into just which pitch-time relations of a tune are fundamental to its particular structure and which are only surface relations--i.e., those which embellish the underlying structure. You can use this knowledge to build variations on a tune or to develop a motive in making a larger piece (see UPS, for instance). Working with TRYS will also help you to hear how composers use some of these same possibilities in more complex pieces to change the character of a motive or to change its function. For example, by keeping the durations the same but changing the pitches, a beginning motive can be transformed into an ending motive while still maintaining its particular identity. Consider, for example, the opening and closing motives of the first movement of the Beethoven 5th Symphony, or, indeed, the transformations which occur throughout the movement. You can experiment with some of these latter possibilities by using TRYPITCH.

II. TRYPITCH :TUNE

TRYPITCH works just like TRYDUR except that you can experiment with new pitches for a given tune while the computer will keep the durations the same. For example:

TRYPITCH "BEETHOVEN
ORIG

This plays the ORIGinal motive: the opening of the 5th Symphony.



> 11 11 11 12 11 11 11 12

This plays the pitches that Beethoven uses at the end of the movement, while keeping the durations the same.

> 15 15 14 12 15 15 14 12

Another set of pitches with the same durations. This transformation occurs right after the opening.

> 11 11 14 14 12 12 15 15

In the transition.

> -1 -1 0 2 -1 -1 0 2

In the transition.

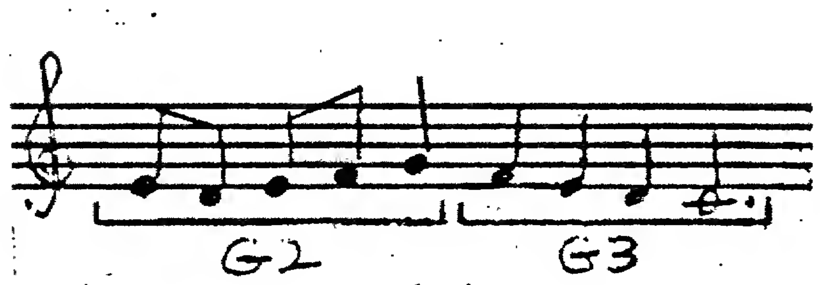
> -5 -5 -5 0 -5 -5 -5 0

As an accompaniment to Theme II.

> 7 7 7 7 10 8 7 5

In the 3rd movement.

III. TRYDUR and TRYPITCH can also be used to discover the "contents" of a tuneblock or whole tune. For example, you may want to <u>reconstruct</u> the actual notes of a particular block. The following is the protocol of someone reconstructing the blocks [G2 G3], let's call him Sid. Sid begins by trying to reconstruct just the durations of [G2 G3]. TRYDUR gives him the pitches of [G2 G3] while he experiments with the durations. (Note: [G2 G3] are two blocks from the tune, FRENCH.)



TRYDUR [G2 G3]

>ORIG	Plays [G2 G3] in its original form.
>4 4 4 4 4 4 4 4 4	Pitches of [G2 G3] each with a duration of 4.
	(Sid notes that the 1st part is too slow.)
>1 1 1 1 1 1 4 4 4	He tries the 1st notes faster (durations of 1) and the last notes slower (duraation of 4).
>ORIG	Sid checks the original to see what the problem is. He hears that the 1st part seems to go too fast and the change to slower notes is in the wrong place.
>2 2 2 2 2 4 4 4 4	He plays the 1st notes twice as slow as in the previous try and also starts the slower notes (4's) sooner.
	He hears that the relation between slow and fast seems right, now, but the fast notes seem to spill over into the slower ones. He concludes that the boundary change must be too late.
>2 2 2 2 4 4 4 4 4	Sid moves the change in duration back 1 note earlier. Sounds good but the last note doesn't last long enough--hard to tell how long when it's the last one.
>ORIG	Sounds pretty much the same.
>PR :DURS	Sid asks the computer to printout the durations (:DURS) for [G2 G3].
>2 2 2 2 4 4 4 4 12	He was almost right--only the last duration is longer. His reconstruction "by ear" was pretty close.
>END	

Now Sid goes to work on the pitches of [G2 G3]. He is told that the pitch collection he will need is 1 2 3 4 and 5--i.e., that is the pitch material of [G2 G3].

TRYPITCH [G2 G3]	
>ORIG	Plays the original [G2 G3].
>END	Sid decides to work on G2, alone.
TRYPITCH [G2]	
>ORIG	Plays original of G2.
>3	Sid hunches that G2 starts about in the middle of the range, so he tries 3 to test that.
>5	He tries pitch 5 on the hunch that the upper boundary of G2 is the top pitch of the range. Both hunches seem to check out.
3 3 4 4 5	Sid is trying to fill in the pitch space between 3 and 5.
3 4 4 4 5	Another way to fill in the pitch space.
>ORIG	Not quite right, yet.
>3 3 3 4 5	Not right, either.
>3 2 4 3 5	Maybe the pitches go down and up to fill in the boundaries. But this sounds really different--he hears groups of 2's--3-2; 4-3; 5. Why?
>ORIG	Still seems more like it goes straight up.
>3 2 3 4 5	That sounds right--it goes <u>around</u> pitch 3 and then straight up.
>ORIG	Yes, sounds the same as the last try.
PR :PITCH	Asks the computer to print out the pitches of G2.
>3 2 3 4 5	Got it!
>END	

Finally, Sid works on the pitches of G3. Seems like it should be easy--G3 just goes straight down.

TRYPITCH [G2 G3]	
>ORIG	Plays original [G2 G3].

>3 2 3 4 5 4 3 2 1	Sid tries the pitches of G2 and adds his hunch for G3--straight down. Sounds right.
>PR :PITCH	
>3 2 3 4 5 4 3 2 1	Got the whole thing!
>END	

It's interesting and useful to note that Sid's strategy is one of working with pitch or time movement within the boundaries of <u>figures</u>. For example, he looks for the change in time (or the change in the pace of motion) which marks the boundary between G2 and G3, and later looks for the pitch motion that fills in the lower and upper pitch boundaries of G2. This strategy is in contrast to the more traditional approach of going one note at a time--next-next-next. It might well be that Sid's strategy--a kind of "top-down" approach--is the more natural of the two. Since we tend to hear structural units, like tuneblocks, before we hear individual notes, we could describe Sid's strategy as one of "intuitive analysis"--<u>from</u> spontaneously perceived "chunks" <u>to</u> their constituent elements.

PROJECTS USING UPS

1. UPS :INT :TIMES

To get used to the idea of using a procedure to make "music," try this simple version of UP. UPS generates a series of rising pitches related to one another by some constant interval (:INT). If :INT is 1, the pitches will go up by an interval of 1; if :INT is 3, the pitches will go up by an interval of 3. You must provide UPS with two inputs--:INT and :TIMES. :TIMES tells UPS how many times to do its thing, i.e., how many notes to make. For example, UPS 1 5 will play 5 (:TIMES) notes, each related to the other by an interval of 1 (:INT)--(0 1 2 3 4). UPS 2 8 will play 8 (:TIMES) notes, each related to the other by an interval of 2 (:INT)--(0 2 4 6 8 10 12 14). When you use UPS, the series of notes will always start at 0 and all notes will have a duration of 4. If you use a negative number for :INT, the series will go down by that interval: UPS -3 4 (0 -3 -6 -9).

Experiment with UPS just enough to get used to using inputs, what they do and especially how they vary the effect of the single procedure, UPS. When you have done some experimenting, write some procedures of your own which use UPS. For example,

```
TO FUN   :INT  :TIMES
10 UPS    1     10
20 UPS    2     10
30 UPS    3     10
END
```

```
TO FUN1
10 REST 2
20 UPS  2  12
30 UPS  3  11
40 UPS  1   9
END
```

TOG "FUN "FUN1

Notice that line 10 in FUN plays a chromatic scale, line 20 plays a whole tone scale and line 30 plays a diminished triad arpeggio. Thus, a single procedure will generate both scales and arpeggios--they are in some important sense "the same."

2. UPSY :INT :DUR :TIMES

This next version adds the possibility of specifying the duration of the notes in a series. For example, UPSY :INT :DUR :TIMES (1 1 13) will play 13 (:TIMES) notes that go up by an interval of 1 (:INT) and each note will have a duration of 1 (:DUR). UPSY 1 2 13 will be the same except each note will have a duration of 2, i.e., it will go twice as slow (take twice as long in Total Time) but cover the same Total Distance with the same number of events. What will happen if you play them together:

CHORUS [[UPSY 1 1 13] [UPSY 1 2 13]]

This new input (:DUR) opens up lots of new possibilities especially if you think in terms of variations of Total Time, Total Distance, Event Time (:DUR), Step Size (:INT) and all the interrelations among them. For example, what happens if you keep the Total Distance the same but vary the Step Size:

UPSY 1 1 25
UPSY 2 1 13
UPSY 3 1 9

What happens if you vary the Event Time and the Step Size but want to keep the Total Time the same? Make some procedures and put them together or just make some one-voice procedures and turn them into a whole piece. Remember you can use a negative number to make a descending series.

3. UP :START :INT :DUR :TIMES

This final version adds the possibility of starting your series on any pitch. For example, UP 5 1 1 10 will play a series of notes that start at pitch 5 (:START), go up by an interval of 1 (:INT), each note will have a duration of 1 (:DUR) and there will be 10 (:TIMES) notes in all. With this added input you can make more interesting melodies:

```
TO TOY
10 UP 5 3 2 4
20 UP 4 3 2 4
30 UP 3 3 2  4
40 UP 2 1 1 15
```

UPs can sometimes create surprises when you listen to them. For example:

```
Example A:  UP -12 1 1 25
Example B:  UP -12 4 1  7
```

You might hear Example B as "faster" than Example A; in fact children often say just that. If you focus on the "boundaries" of each example, then, indeed, Example B covers the same pitch distance (2 octaves) as Example A, but it does so in much less time. It's quite natural to have the impression of "going faster" if you get from here to there in less time than you did before. But if you focus on the rate from one event to the next, the two examples are, of course, the same in "fastness." Composers make use of these kinds of shifts in focus to change the structural rhythm of a piece while keeping the surface durations the same. Listen, for instance, to Vivaldi, The Seasons, "Winter."

You can also discover some interesting aspects of musical structure by using UPs. For example, UPs will make a CHROMATIC scale:

```
UP -12 1 2 25
```

or a whole-tone scale:

```
UP -12 2 2 13
```

or a diminished triad arpeggio:

```
UP -12 3 2 9
```

or an augmented triad arpeggio:

```
UP -12 4 2 7
```

but UP cannot make a MAJOR scale. Why not?

UP and DOWN can also be used to build tunes and even to build whole pieces. You can, for example, make a procedure that plays a tune which is itself made up of UPs:

```
TO LINE1
10 UP 0 2 2 5
20 REST 6
30 UP 1 2 2 5
40 REST 6
50 UP 2 2 2 4
60 DOWN 10 2 2 4
70 DOWN 4 1 2 5
80 REST 6
END
```

```
TO LINE2
10 REST 8
20 DOWN 16 2 2 5
30 REST 6
40 DOWN 15 2 2 5
50 REST 6
60 DOWN 14 2 2 4
70 UP 6 1 2 5
80 REST 6
END
```

These two procedures can then be used as modules and played together in a CHORUS:

```
CHORUS [[LINE1] [LINE2]]
```

Other such modules can be built and played together or separately, each one maintaining its integrity but still transforming as it is embedded in changing "sound environments." For example:

```
TO LINE3
10 DOWN 8 3 4 4
20 DOWN 6 3 4 4
30 DOWN 4 3 4 4
40 DOWN 2 3 4 4
50 END
```

Each of the larger modules (LINE1, LINE2, LINE3) can become a module in a superprocedure which we'll call MYPIECE. Notice that as you superimpose the larger modules in various combinations, the superprocedure, as description, captures your immediate apprehension of increasing density and activity of texture as well as the particular relations among separate voices. You can *hear* what the notation describes.

```
   TO MYPIECE
10 LINE1
20 CHORUS [LINE1 LINE2]
30 CHORUS [LINE1 LINE3]
40 CHORUS [LINE1 LINE2 LINE3]
50 CHORUS [LINE2 LINE3]
60 LINE1
   END
```

CANONS

A CANON (or round) is a piece for several instruments or singers, each of whom plays or sings exactly the same melody. However, each performer begins at a different time so that we, in fact, hear the melody played against itself. You are probably familiar with "Row, Row, Row Your Boat," which works as a canon. It is easy to make the computer play a melody "in canon"--i.e., in several voices staggered in their entrances. The game is to figure out just when each part should enter. In fact, this is a game composers have been playing since the time of Bach. Mozart, for instance, wrote many canons which he often sent to friends at the end of a letter as a "riddle." The game was just the same as the one you can play using the computer--namely, to figure out where each voice should start so that the melody will sound well against itself. You can use ROW as a first try.

You: ROW — This plays the tune, Row.

Listen to the tune and try to guess when the second voice should enter. Let's say that you want the second voice to begin after the word, "boat"--i.e., on the word "gently." Listen to the tune again and count the number of beats up to the word "gently." You should have counted 4 beats. The unit beat for ROW is 4, so 4 beats will be 16 computer counts (C.C.). That means you should ask the computer to wait 16 C.C. before beginning the second voice.

You: CANON "ROW 16 2 — This tells the computer to play the tune ROW, then after 12 C.C., play ROW in the second voice. There will be only 2 voices in this first try.

Voice 1: |———— ROW ————|

Voice 2: 16 C.C. |———— ROW ————|

You: CANON "ROW 16 4

This will play the canon, ROW in 4 voices. Each voice will start 12 C.C.'s after the one preceding it.

```
Voice 1: |---------------- ROW ----------------|
Voice 2: 16
         C.C. |---------------- ROW ----------------|
Voice 3:          16
                  C.C. |---------------- ROW ----------------|
Voice 4:                   16
                           C.C. |---------------- ROW ----------------|
```

SIMILAR PROCEDURES

IN

DIFFERENT MEDIUMS

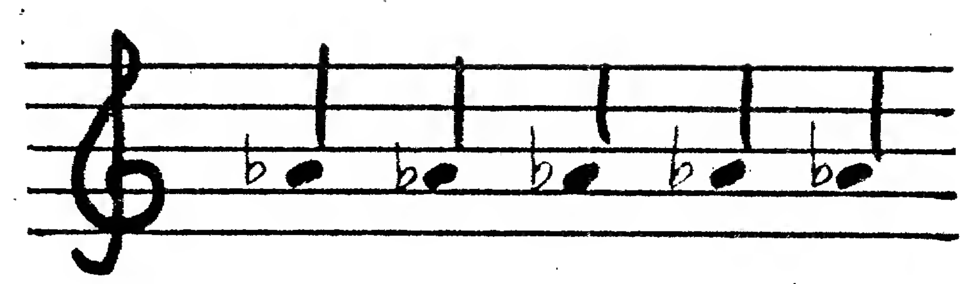
```
TO REPNUM :NUM :TIMES
10 IF :TIMES = 0 STOP
20 PRINT :NUM
30 REPNUM :NUM :TIMES-1
END
```

```
REPNUM 10 5
     10
     10
     10
     10
     10
```

***

```
TO REPPITCH :PITCH :TIMES
10 IF :TIMES = 0 STOP
20 PLAY :PITCH 4
30 REPPITCH :PITCH :TIMES-1
END
```

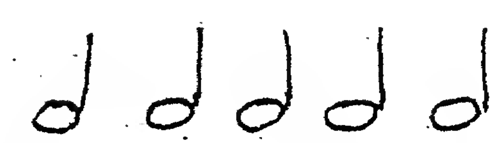
REPPITCH 10 5

PLAYS: 

***

```
TO BEAT :DUR :TIMES
10 IF :TIMES = 0 STOP
20 BOOM  :DUR
30 BEAT :DUR :TIMES-1
END
```

BEAT 8 5

PLAYS: 

(on the computer drum)

***

```
TO REPSQ :SIDE :TIMES
10 IF TIMES = 0 STOP
20 MAKE ''A HEADING
30 FD :SIDE RT 90 UNTIL HEADING = :A
40 REPSQ :SIDE :TIMES-1
END
```

REPSQ 100 4

(Makes a square of side 100 and flashes it 4 times.)

(35 WAIT 10 CS WAIT 50) = BUG

***

```
TO DOWNNUM :NUM :DECRE
10 IF :NUM < -24 STOP
20 PRINT :NUM
30 DOWNNUM :NUM-:DECRE :DECRE
END
```

```
DOWNNUM 10 10
 10
  0
-10
-20
```

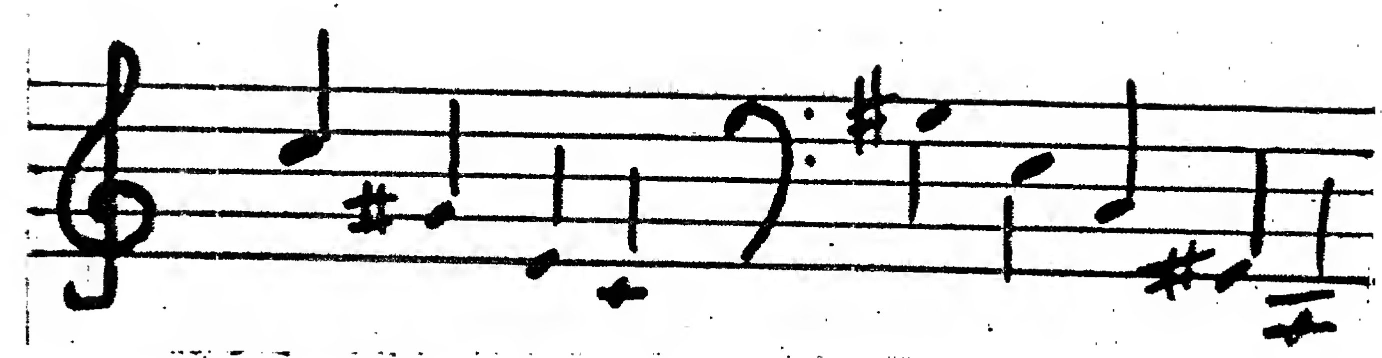
```
DOWNNUM 100 50
100
50
0
```

```
DOWNNUM 10,000 50
 10,000
  9,950
  9,900
  9,850
  9,800
  9,750
```

***

```
TO DOWNPITCH :PITCH :DECRE
10 IF :PITCH <-24 STOP
20 PLAY :PITCH 4
30 DOWNPITCH :PITCH-:DECRE :DECRE
END
```

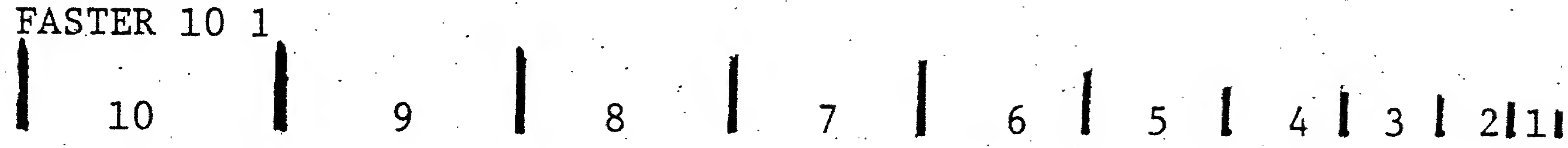
DOWNPITCH 12 4



***

```
TO FASTER :DUR :DECRE
10 IF :DUR < 1 STOP
20 BOOM :DUR
30 FASTER :DUR-:DECRE :DECRE
END
```

FASTER 10 1

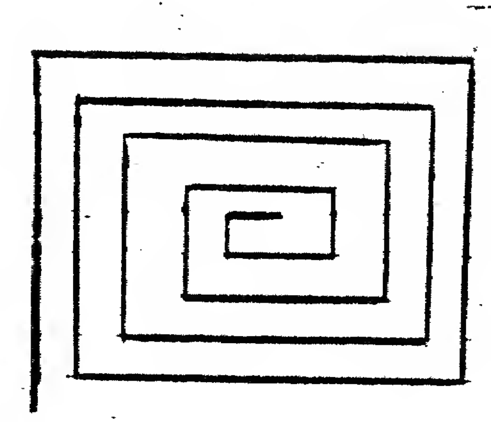


(BOOM [10-9-8-7-6-5-4-3-2-1])

***

```
TO SQUARAL :SIDE :DECRE
10 IF :SIDE<1 STOP
20 FD :SIDE RT 90
30 SQUARAL :SIDE-:DECRE :DECRE
END
```

SQUARAL 100 10



***

MUSIC PROJECTS

I. Beginning Projects

1. Building a tune with pre-programmed "tuneblocks" (small, melodic motives).
2. Building drum pieces with pre-programmed "rhythm blocks."
3. Using drum pieces as accompaniments to tunes.
4. Building interesting rhythms--learning to measure time.
5. Making special kinds of tunes--scary tunes, funny tunes, "good" tunes--by combining tuneblocks in particular ways.
6. Using and building tune modules as parts of bigger pieces.

II. Intermediate Projects

1. Inventing repeating melodic patterns to accompany tunes in the bass.
2. Building tunes (familiar and new ones) by specifying each pitch and duration.
3. Making a set of variations on a theme by manipulating pitch or time separately or by adding different accompaniments.
4. Using procedures with several inputs to explore relations between pitch, time, "distance," and number, for example:

```
UP :START :INTERVAL :DUR :TIMES
```

5. Write procedures to:
   a. generate a steady pulse at variable rates.
   b. make a rhythm that gets faster or slower.
   c. make a series of pitches that gets higher or lower at varying rates.
   d. make a series of pitches that gets both higher/lower and faster/slower.

III. Advanced Projects

1. Write and use procedures that will:
   a. generate chords which will harmonize tunes.
   b. "compose" pieces on a structural level--i.e., that will describe structural relations of a piece, not just notes.
   c. coordinate the rhythm of a tune with the movements of the turtle.
   d. coordinate the "shape" of a tune with a visual shape drawn by the turtle--i.e., play the tune and draw a picture.
   e. generate random strings of pitches and/or durations from which to design and develop pieces.
   f. generate random arrangements of tuneblocks to test what we mean by

a good or a bad tune.

g. generate <u>good</u> arrangements of tuneblocks based on tests made, above.

h. generate <u>good</u> arrangements of pitches and durations based on the decisions used in 'e' above.